tt: Treelet transform with Stata

Gorst-Rasmussen, Anders

Publication date:
2011

Document Version
Early version, also known as pre-print

AALBORG UNIVERSITY

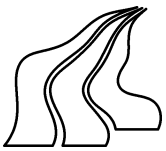
tt: Treelet transform with Stata

by

Anders Gorst-Rasmussen

R-2011-09 July 2011

Department of Mathematical Sciences
Aalborg University
Fredrik Bajers Vej 7 G ▪ DK - 9220 Aalborg Øst ▪ Denmark
Phone: +45 99 40 80 80 ▪ Telefax: +45 98 15 81 29
URL: http://www.math.aau.dk



ISSN 1399–2503 ▪ On-line version ISSN 1601–7811

Submitted to 'Stata Journal'

tt: Treelet transform with Stata

Anders Gorst-Rasmussen
Department of Mathematical Sciences
Aalborg University
gorst@math.aau.dk

July 4, 2011

Abstract. The treelet transform (TT) is a recent data reduction technique from the field of machine learning. Sharing many similarities with principal components analysis (PCA), TT can reduce a multidimensional data set to the projections on a small number of directions or components which account for much of the variation in the original data. However, in contrast to PCA, TT produces sparse components. This can greatly simplify interpretation. We describe the `tt` Stata add-on for performing TT. The add-on includes a Mata implementation of the TT algorithm, alongside other functionality to aid the practical application of TT. We show how a basic exploratory data analysis using the `tt` add-on might look.

Keywords: treelet, PCA, dimension reduction, factor analysis.

1 Introduction

A common task in data analysis is to summarize a multidimensional data set. One popular and convenient approach is to find a few interesting directions in the data and use the corresponding linear projections of data as representatives of the original data in plots, regression models etc. This is known as dimension reduction. Principal components analysis (PCA) is a standard dimension reduction method which works by calculating the first few eigenvectors (components) of a covariance or correlation matrix and reducing the data set to a collection of component scores – the projection of data onto components. This strategy has the optimality property of explaining as much variation as possible in the original data using as few dimensions as possible. Often, entries of the components (loadings) are subjected to interpretation. Variables corresponding to 'large' loadings are interpreted as being important for describing the original data; variables corresponding to 'small' loadings can be discarded. Such interpretation is complicated by the fact that all component loadings are nonzero. Various cutoff rules, component rotation strategies etc. have been developed to simplify interpretation (Jolliffe, 2002) but these largely ad hoc procedures do not contribute to the transparency and objectivity of PCA.

In the machine learning community, there has been a growing interest in developing alternatives to PCA which offer more interpretable components by forcing loading patterns where many loadings

are exactly zero, i.e. by forcing sparse components. For example, Zou et al. (2006) developed a variant of PCA where sparse components are estimated via penalized regression with automatic variable selection. The treelet transform (TT) proposed by Lee et al. (2008) is a similar recent alternative to PCA. TT introduces sparsity among component loadings in an elegant and simple fashion by combining ideas from hierarchical clustering analysis with ideas from PCA. This leads to sparse components which, similarly to PCA components, account for a large part of the variation in the original data and can be used in an analogous manner. In addition, it leads to an associated cluster tree which provides a concise visual representation of loading sparsity patterns and the general dependency structure of the data.

We describe in this paper the Stata add-on `tt` (Gorst-Rasmussen, 2011) which contains a Mata implementation of the TT algorithm. In addition to the TT algorithm itself, `tt` includes a number of other functions to aid in model selection and output analysis in practice. Using the `cars` data set which comes with Stata, we provide a small demonstration of how the various functions work together, and how a complete TT analysis using `tt` might look.

2 The treelet transform algorithm

This section provides a brief, nontechnical review of the TT algorithm. For a more formal derivation of TT and its properties, see the original paper by Lee et al. (2008).

Given a collection of p variables, the TT algorithm proceeds as follows:

Variable pairing. Locate the two variables with the largest correlation coefficient.

Local PCA. Merge these two variables by performing PCA on them. Keep the new variable/score with the largest variance (the 'sum' variable), discard the other new variable/score (the 'residual' variable).

This yields a new collection of $p-1$ variables, namely the sum variable and the remaining $p-2$ original variables, on which we then repeat the above two steps. The 'variable pairing'/'local PCA' scheme is repeated for a total of $p-1$ times until only a single sum variable is left. This in turn defines a basic hierarchical clustering algorithm, the output of which is conveniently represented as a binary tree with p levels (a cluster tree or cluster dendrogram). Variables that are 'close' in this cluster tree, and are merged early, represent groups of more highly correlated variables.

Hierarchical clustering is in itself a well-known technique. The novelty of TT is its use of PCA to merge variables since it enables us to construct, at each level of the TT cluster tree, a complete coordinate system for the data. Specifically, viewing TT in terms of its action on components rather than variables, we start out with a coordinate system consisting of the trivial, one-variable components (the standard coordinate system of $\mathbb{R}^p$). Each local PCA of two variables corresponds to performing an orthogonal rotation of two components. It follows that a coordinate system for the data at a given level of the TT cluster tree is given by the collection of:

1. The components corresponding to sum variables available at the current level and;
2. components corresponding to all previously calculated residual variables and;
3. 'trivial' components for variables that have not yet joined the cluster tree.

The level- and data-specific coordinate system is thus comprised of 'sum' components which encode coarse-grained, low-resolution information about the dependency relationships between all variables included so far; alongside 'residual' components which encode information about the more local relationships between variables at an increasingly greater resolution. It can be shown that if TT is applied to a collection of variables with a covariance matrix featuring high intrablock correlation and low interblock correlation then the loadings of sum components will be constant on variables within blocks (Lee et al., 2008) in large samples. Hence, TT can help identify groups of correlated variables.

2.1 Selecting a cut-level

Application of TT to a data set yields, as its basic output, a cluster tree alongside a coordinate system for the data at each level of the cluster tree. As described above, the coordinate system combines coarse components not unlike components obtained from PCA, with higher-resolution components which reflect local dependency relationships. We seek to utilize this collection of coordinate systems for dimension reduction purposes.

If we knew which cluster tree level (cut-level) to use, we could calculate variances of the level-specific component scores and retain components corresponding to the highest-variance scores. This is the approach used in PCA with one difference: TT component scores are generally correlated and do not lead to a true decomposition of variance. This is a known issue in dimension reduction (Gervini and Rousson, 2004) since PCA is the only method yielding both orthogonal components and uncorrelated scores.

Selecting a cut-level for the TT cluster tree amounts to deciding the level of detail desired in the dimension reduction, i.e. the amount of regularization. A coordinate system close to the leaves of the cluster tree contains mostly highly sparse components and may not be useful for dimension reduction in the sense that the high-resolution components are not much more informative than the original one-variable components. Conversely, a coordinate system close to the root includes coarse-grained, low-resolution components more suitable for dimension reduction but may be harder to interpret because of lacking sparsity. We usually prefer a data-driven choice of cut-level. Choosing a cut-level from data is not trivial since coordinate systems at different cut-levels are equally capable of describing the data if only we use a sufficiently large number of components. However, cross-validation methods can be used to find a cut-level at which we can describe the data using only few components. Suppose that we wish to describe the data using exactly m components. Then we determine an appropriate cut-level by using the following K-fold cross-validation strategy (Lee *et al.* (2008)):

1. Split the data randomly into K roughly equal-sized subsets. For each of these subsets, do the following:

    - For each cut-level $1, \ldots, p-1$ calculate the m highest-variance components using all subsets of data *except* the current. Next, calculate the sum of variances of scores based on these components using only the *current* subset.

2. For each cut-level $1, \ldots, p-1$, calculate a cross-validation score by averaging the K sums of component variances obtained in step 1.

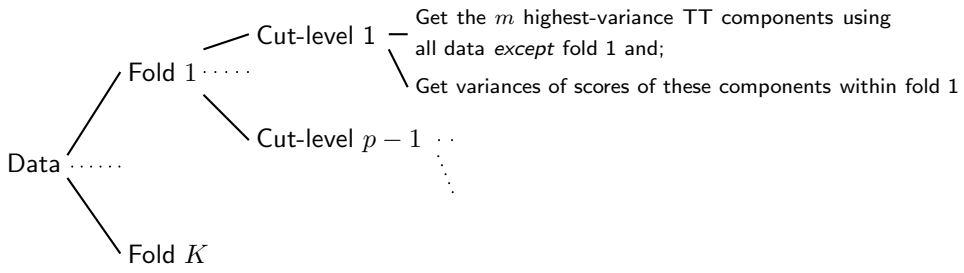


Figure 1: Flow chart for the cross-validation strategy for deciding an optimal cut-level.

A flowchart visualizing step 1 of the cross-validation strategy is shown in Figure 1.

Once cross-validation scores have been obtained, a suitable cut-level can be found by locating a 'knee' on the graph of cross-validation scores against cut-level, i.e. a point at which increasing the cut-level does not substantially increase the cross-validation score. In other words, we select the cut-level at which we can explain almost as much variation as possible, using as low a cut-level as possible to simplify interpretation of components.

Note that the cross-validation strategy requires us to specify the number of components m to use. This is not much different from the corresponding problem of selecting the number of components to retain in PCA; or the number of clusters in a cluster analysis. In Section 4, we propose a simple data-driven strategy for selecting both cut-level and the number of components.

2.2 Stability assessment

A data analyst may wish to know how much trust to place in a collection of components obtained using TT. Since a key feature of TT is its ability to produce sparse components, it is of particular interest to assess the stability of loading sparsity patterns. This can be done by using a subsampling approach inspired by Ben-Hur et al. (2002).

We first specify a cut-level k and a number m of TT components to retain. Then we repeat the following subsampling scheme 100 times:

1. Randomly sample 80% of the data.
2. Within this subsample, calculate the m highest-variance TT components at cut-level k of the cluster tree. For each of these m components, do the following:
    - Calculate the sign pattern of the component. For example, a component $(-0.1, 0.2, 0, 0.1)$ corresponds to the sign pattern $(-, +, 0, +)$.
    - Calculate the variance explained by the corresponding component.
    - Calculate the rank according to the variance explained by the corresponding component.

The collection of all $100 \cdot m$ sign patterns, alongside their variances and ranks, carries information about the stability and the importance of different sign patterns appearing in the subsampled TT analyses. As a measure of stability, we count the number of times we see a particular sign pattern

among all $100 \cdot m$ patterns while using the average rank and average variance of the sign pattern as measures of importance. The final output of the stability analysis is the relative frequency, average variance, and average rank of each sign pattern occurring in more than 10 out of the 100 subsampled TT analyses. Note that this number is generally different from m.

3 The `tt` add-on

3.1 Syntax

The main function in the `tt` add-on (Gorst-Rasmussen, 2011) is implemented as a Mata function called via a Stata wrapper. It is loosely based on the R-code by Liu (2010) and has syntax:

`tt` *varlist*[*if*] [*in*] [*weight*] `, cut(#)` [`options`]

After calling `tt`, the user will typically call `ttcv` which uses the cross-validation strategy of Section 2.1 to select a cut-level for the TT cluster tree. It has the following syntax:

`ttcv` *varlist*[*if*] [*in*] [*weight*] `, components(#)` [`options`]

A range of different post-estimation commands is also available. As usual with post-estimation commands, they require an initial call to `tt`. Stability assessment as described in Section 2.2 is available in the command `ttstab` which has syntax:

`ttstab` [`, options`]

The TT cluster tree can be plotted by using the following command:

`ttdendro` [`, dendro_options`]

Scree plot of variances and 'skyscraper plots' of component loadings are implemented in the commands `ttscree` and `ttloading`, respectively, with syntax:

`ttscree` [`, options scatter_options`]

`ttloading` [`, options scatter_options`]

Finally, `ttpredict` implements prediction of component scores. As previously described, these are the projections of the original data onto the relevant TT component and can be informally interpreted as the degree of 'adherence' of a given observation vector to the given component. The `ttpredict` syntax is:

`ttpredict` [*if*] [*in*] `{stub*|newvarlist}`

3.2 tt options

`cut(#)` is required and specifies the cut-level of the TT cluster tree at which to extract components. The cut-level influences both the sparsity and composition of components.

`components(#)` sets the maximum number of components to be retained. `tt` displays the full set of components variances but displays loadings only for retained components. The default is the

number of variables in *varlist*.

corrrelation or covariance specifies that TT cross-validation be based based on the correlation matrix or covariance matrix, respectively. The default is correlation. Usually, TT based on the covariance matrix will be meaningful only if variables are expressed in the same units.

noblanks display zero loadings as 0 instead of blanks; included for readability.

3.3 ttcv options

components(#) is required and sets the number of components to be retained. In practice, this number may not be known in advance; in which case one should investigate the output of ttcv for a range of different choices components().

folds(#) specifies the number of folds (test samples) to use in cross-validation. The default is folds(10).

reps(#) specifies the number of Monte-Carlo repetions of cross-validation. Default is reps(5). Monte-Carlo repetitions reduce the sampling variation inherent in cross-validation; increase reps(#) if the output of ttcv appears unstable over different runs.

percent(#) specifies that a "knee" on the graph of cross-validation scores should be sought among cut-levels for which the score is within #percent of the cross-validation score associated with the maximal cut-level. Default is percent(10).

corrrelation or covariance specifies that TT cross-validation be based based on the correlation matrix or covariance matrix, respectively. The default is correlation. Usually, TT based on the covariance matrix will be meaningful only if variables are expressed in the same units.

force try to force cross-validation even when zero-variance variables are detected in training samples. This is usually an indication that there is something wrong; use this option with caution.

3.4 ttstab options

reps(#) number of subsamples; default is reps(100).

subsample(#) subsample size in percent of the original sample size; default is subsample(80).

keep(#) keep sign patterns appearing in more than # percent of replications; default is keep(20).

force tries to force subsampling even when zero-variance variables are found in subsamples. This is usually an indication that there is something wrong; use this option with caution.

3.5 ttdendro options

dendro_options are any of the options allowed by the cluster dendrogram command; see [MV] **cluster dendrogram**.

3.6 ttscree and ttloading options

scatter_options are any of the options allowed by the graph twoway scatter command; see [G] **graph twoway scatter**.

The following option applies only to `ttscree`:

`neigen` plot only the largest first # component variances; default is to plot all component variances

The following option applies to `ttloading` only:

`components` plot components in *numlist*; default is `components(1 2 3)`.

4 A data example

As a simple illustration of the proposed workflow when using the `tt` add-on, we consider the 1978 automobile toy data set which comes with Stata. This data set describes various characteristics of a total of 74 vehicles. We will use the 10 variables described below for the analysis; a total of 69 vehicles have complete observations for these variables.

```
. sysuse auto
(1978 Automobile Data)
. describe price-gear_ratio

              storage  display     value
variable name   type   format      label      variable label

------------------------------------------------------------------------------
price           int    %8.0gc                 Price
mpg             int    %8.0g                  Mileage (mpg)
rep78           int    %8.0g                  Repair Record 1978
headroom        float  %6.1f                  Headroom (in.)
trunk           int    %8.0g                  Trunk space (cu. ft.)
weight          int    %8.0gc                 Weight (lbs.)
length          int    %8.0g                  Length (in.)
turn            int    %8.0g                  Turn Circle (ft.)
displacement    int    %8.0g                  Displacement (cu. in.)
gear_ratio      float  %6.2f                  Gear Ratio
```

4.1 Step 1: running tt

To get familiar with the data set, we first make a couple of preliminary runs of `tt` and the `tt_postestimation` plotting routines.

```
. tt price-gear_ratio, cor cut(3) components(3)

Treelet transform/correlation                   Number of obs    =        69
                                                Number of comp.  =         3
                                                Cut-level        =         3
    ------------------------------------------------------------------------------
       Component |     Variance   Proportion   Cumulative    Adj. proportion
    -------------+----------------------------------------------------------------
             TC1 |       3.6404       0.3640       0.3640             0.3640
             TC2 |       1.0000       0.1000       0.4640             0.0360
             TC3 |       1.0000       0.1000       0.5640             0.0746
             TC4 |       1.0000       0.1000       0.6640             0.0344
             TC5 |       1.0000       0.1000       0.7640             0.0787
             TC6 |       1.0000       0.1000       0.8640             0.0371
             TC7 |       1.0000       0.1000       0.9640             0.0652
             TC8 |       0.1875       0.0187       0.9828             0.0143
             TC9 |       0.1199       0.0120       0.9948             0.0086
            TC10 |       0.0522       0.0052       1.0000             0.0031
    ------------------------------------------------------------------------------
```

```
Components

    ------------------------------------------
        Variable |      TC1       TC2       TC3
    -------------+----------------------------
           price |
             mpg |
           rep78 |
        headroom |                        1.0000
           trunk |
          weight |   0.5080
          length |   0.5080
            turn |   0.4851
    displacement |   0.4985
      gear_ratio |             1.0000
    ------------------------------------------

. tt price-gear_ratio, cor cut(6) components(3)

Treelet transform/correlation                    Number of obs    =        69
                                                 Number of comp.  =         3
                                                 Cut-level        =         6
    ------------------------------------------------------------------------------
       Component |     Variance   Proportion   Cumulative    Adj. proportion
    -------------+----------------------------------------------------------------
             TC1 |       4.5497       0.4550       0.4550             0.4550
             TC2 |       1.6565       0.1657       0.6206             0.0432
             TC3 |       1.0000       0.1000       0.7206             0.0800
             TC4 |       1.0000       0.1000       0.8206             0.0717
             TC5 |       0.6353       0.0635       0.8842             0.0515
             TC6 |       0.4555       0.0455       0.9297             0.0328
             TC7 |       0.3435       0.0343       0.9640             0.0335
             TC8 |       0.1875       0.0187       0.9828             0.0143
             TC9 |       0.1199       0.0120       0.9948             0.0086
            TC10 |       0.0522       0.0052       1.0000             0.0031
    ------------------------------------------------------------------------------

Components

    ------------------------------------------
        Variable |      TC1       TC2       TC3
    -------------+----------------------------
           price |
             mpg |             0.7071
           rep78 |                        1.0000
        headroom |   0.3052
           trunk |   0.3639
          weight |   0.4471
          length |   0.4471
            turn |   0.4269
    displacement |   0.4387
      gear_ratio |             0.7071
    ------------------------------------------

. ttdendro
. ttscree
```

In both calls to `tt`, we retain 3 components but use different cut-levels 3 and 6, respectively. The relatively low cut-level of 3 in the first analysis yields more sparse components. In fact, components 2 and 3 in this first analysis are somewhat uninteresting for the purpose of dimension reduction since they contain only a single variable. The second analysis uses the cut-level 6 and leads to less sparse components.

The call to `tt` returns both the 'raw' variances explained by components and variances adjusted for correlation between scores using the conservative method of Gervini and Rousson (2004). For

the present data, the first TT component explains the majority of the variation for both cut-levels 3 and 6, irrespective of the method used for variance calculation. In both analyses, this first component can be informally interpreted as measuring the overall 'size' of a vehicle.

The output of the call to `ttdendro` is shown in Figure 2. The TT cluster tree shows that `trunk`, `weight`, `length`, `displacement`, and `turn` form a tight cluster. With the addition of the variable `headroom`, it is this particular cluster that is reflected by the first TT component in the second call to `tt` above. It is a general feature of the TT algorithm that cluster membership in the cluster tree translates to nonzero loadings in some TT component. In other words, the cluster tree provides a concise visual representation of the possible TT components.

Figure 3 is obtained by calling `ttscree`. It is a graphical representation, similar to PCA scree plots, of the (unadjusted) variance explained by components. It is clear from this plot that a single component suffices to capture much of the variation in the data.

The first TT component in the second call to `tt` above is very similar to the first component obtained from the corresponding PCA, as can be seen from the numerical loadings and Pearson correlation between scores calculated below. However, the first TT component is potentially simpler to interpret because of its sparsity.

```
. ttpredict tt1score
(9 components skipped)
. pca price-gear_ratio, cor components(2)

Principal components/correlation                  Number of obs    =         69
                                                  Number of comp.  =          2
                                                  Trace            =         10
    Rotation: (unrotated = principal)             Rho              =     0.7389

    --------------------------------------------------------------------------
       Component |   Eigenvalue   Difference         Proportion   Cumulative
    -------------+------------------------------------------------------------
           Comp1 |      6.31248      5.23618             0.6312       0.6312
           Comp2 |       1.0763     .0622654             0.1076       0.7389
           Comp3 |      1.01403      .583752             0.1014       0.8403
           Comp4 |      .430283     .0343745             0.0430       0.8833
           Comp5 |      .395908      .116712             0.0396       0.9229
           Comp6 |      .279196     .0229213             0.0279       0.9508
           Comp7 |      .256275      .130573             0.0256       0.9764
           Comp8 |      .125701     .0442338             0.0126       0.9890
           Comp9 |     .0814675     .0531123             0.0081       0.9972
          Comp10 |     .0283551            .             0.0028       1.0000
    --------------------------------------------------------------------------

Principal components (eigenvectors)

    ------------------------------------------------
        Variable |    Comp1     Comp2 | Unexplained
    -------------+--------------------+-------------
           price |   0.2074    0.3876 |       .5668
             mpg |  -0.3394    0.0520 |       .2699
           rep78 |  -0.1830    0.7639 |       .1606
        headroom |   0.2304    0.3049 |        .565
           trunk |   0.3003    0.3401 |       .3061
          weight |   0.3848    0.0095 |      .06535
          length |   0.3771    0.0432 |       .1003
            turn |   0.3542   -0.1831 |       .1719
    displacement |   0.3742   -0.0121 |       .1157
      gear_ratio |  -0.3306    0.1388 |       .2895
    ------------------------------------------------

. predict pc1score
```

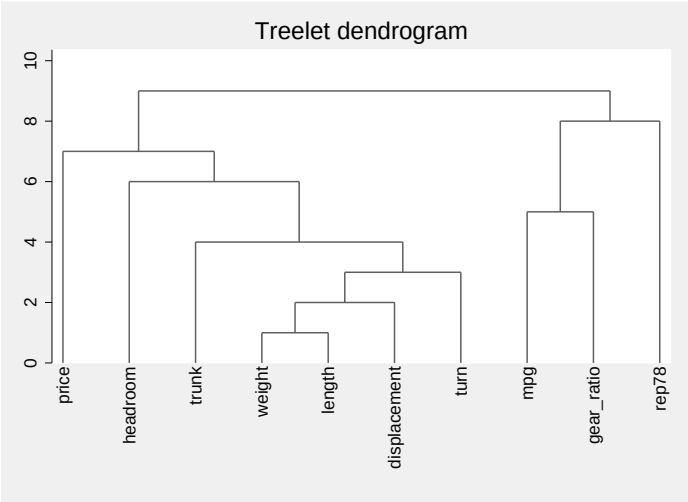


Figure 2: Cluster tree produced by `tt`.

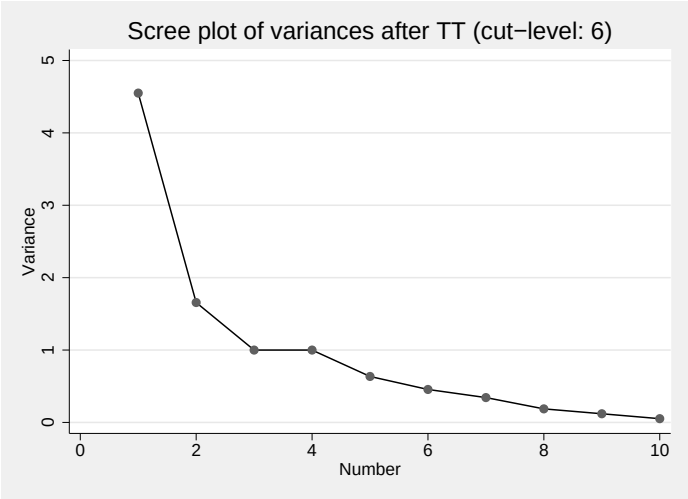


Figure 3: Scree plot of variances of TT component scores when the cut-level 6 is used.

```
. correlate tt1score pc1score
(obs=69)
             | tt1score pc1score
-------------+------------------
    tt1score |   1.0000
    pc1score |   0.9842   1.0000
```

4.2 Step 2: running ttcv

From the analysis in step 1, we found evidence that a single TT component suffices to describe the majority of variation in the data. It turns out that the optimal cut-level for a single-component solution is 9, the maximal possible; and that the single retained component has all nonzero loadings for this cut-level.

For the purpose of illustration, suppose instead that we decide to keep 3 components. We can then find a suitable cut-level by a call to `ttcv` as follows.

```
. ttcv price-gear_ratio, cor components(3)

 Cross-validation (10 folds, 5 repetitions)
0%            25%            50%            75%          100%
+--------------+--------------+--------------+--------------+
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

TT cross-validation/correlation        Number of obs   =     69
                                       Number of comp. =      3

                                       Number of folds =     10
                                       Number of reps  =      5

Cross-validation scores
    ------------------------------------------
       Cut-level |       Score    Proportion
    -------------+----------------------------
                1 |       5.3364       0.6509
                2 |       6.1585       0.7512
                3 |       6.9091       0.8428
                4 |       7.2181       0.8805
                5 |       7.5062       0.9156
                6 |       7.8466       0.9571
                7 |       7.7807       0.9491
                8 |       7.9603       0.9710
                9 |       8.1980       1.0000
    ------------------------------------------
    Estimated optimal cut-level = 6
    (optimal cut-level sought within 10% of highest cut-level score)
```

Figure 4 shows a plot of the cross-validation scores generated when calling `ttcv`. Although not entirely convincing, a 'knee' in the graph seems to be located around level 6, indicating that increasing the cut-level beyond this level will not substantially improve the amount of variance explained by the 3 components. Thus, for a 3-component solution, a cut-level of 6 appears adequate.

Choosing simultaneously the number of components to retain *and* a cut-level is easy for the present data set since a single component-solution seems to be preferable at most nontrivial cut-levels. In situations where it is unclear how many components to retain, the choice can be more difficult. The following strategy is recommended:

- Decide on a range of different sensible values of `components()` in the call to `tt` via, for

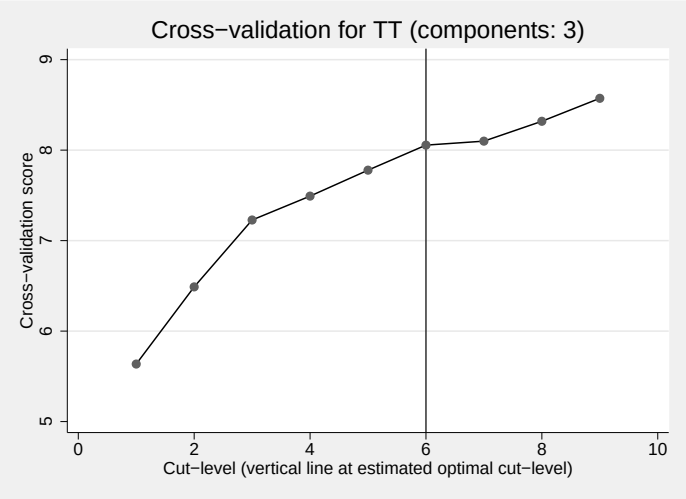


Figure 4: Graph of cross-validation scores for TT when 3 components are retained. The graph suggests that a 'knee' in the graph is located at cut-level 6.

example, investigation of scree plots.

- Perform `ttcv` for each of these choices of `components()`.

In our experience, there will often be a reasonably small range of cut-levels that are universally preferable for the selected range of `components()`. A parsimonious solution is then to use the smallest acceptable cut-level among these.

4.3 Step 3: running ttstab

For the choice `cut(6)` and `components(3)` in the call to `tt`, we conclude our analysis by investigating the stability of the obtained solution via a call to `ttstab`.

```
. tt price-gear_ratio, cor cut(6) components(3)
  (output omitted)
. ttstab

Subsampling repetitions (100)
0%           25%           50%           75%          100%
+-------------+-------------+-------------+-------------+
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


Stability of TT/correlation                   Number of obs    =        69
                                              Number of comp.  =         3
                                              Cut-level        =         6

                                              Subsample rep.   =       100
                                              Subsample frac.  =      0.80
                                              Subsample size   =        55

Average rank (by amount of variance explained) and frequency of sign patterns
```

```
Displaying results for patterns with frequency >=  10%

------------------------------------------------------------
Sign pattern |   Avg. rank   Frequency   Avg. variance
-------------+----------------------------------------------
           1 |       1.000       0.890           4.552
           2 |       2.000       0.990           1.656
           3 |       3.000       0.350           1.000
           4 |       3.000       0.610           1.000
------------------------------------------------------------

Structure of sign patterns

-------------------------------------
    Variable |    1    2    3    4
-------------+-----------------------
       price |    0    0    +    0
         mpg |    0    +    0    0
       rep78 |    0    0    0    +
    headroom |    +    0    0    0
       trunk |    +    0    0    0
      weight |    +    0    0    0
      length |    +    0    0    0
        turn |    +    0    0    0
displacement |    +    0    0    0
  gear_ratio |    0    +    0    0
-------------------------------------
```

The call to `ttstab` performs 100 subsampling repetitions of TT, keeping the 3 highest-variance components in each subsampled analysis (at cut-level 6). It then transforms these into their corresponding sign patterns. Note that `ttstab` is set to return all sign patterns seen in more than 10% of the subsampling repetitions, here corresponding to 4 sign patterns. In the output, 'Avg. rank' is the the rank (according to explained variance of the corresponding component), averaged over the 100 subsamples. 'Frequency' is the relative frequency of the sign pattern among all $3 \cdot 100$ sign patterns returned. Lastly, 'Avg. variance' is the variance explained by the component corresponding to the sign pattern, averaged over the 100 subsamples.

We can see that sign patterns similar to those of the first two components from the original TT analysis with `components(3` and `cut(6)` appear in almost all subsampling repetitions. If the first type of sign pattern appears, it corresponds to a component with rank 1. Moreover, the first component remains by far the most important one in terms of variance explained. Sign patterns 3 and 4, on the other hand, do not appear to be very stable. Increasing the number of retained components to 4 does lead to a greater stability in terms of frequency of inclusion but does not improve stability of the rank of the last two components.

5 Concluding remarks

The treelet transform can be viewed as an amalgamation of PCA and cluster analysis. It leads to components that are sparse and can be easier to interpret than their PCA counterparts. We have described the `tt` add-on for Stata which contains all the basic functionality needed to apply the treelet transform in practice, including an Mata implementation of the treelet transform algorithm. For a more advanced application example and a detailed comparison with the output produced by PCA, we refer to the paper by Gorst-Rasmussen et al. (2011).

6 Acknowledgements

I thank Søren Lundbye-Christensen and Christina C. Dahm for their helpful comments and suggestions when preparing this manuscript. Part of the development work on the `tt` add-on was completed while on a leave from Aalborg University and employed at Centre for Cardiovascular Research, Aalborg Hospital, Aarhus University Hospital.

References

Ben-Hur, A., A. A. Elisseeff, and I. Guyon. 2002. A stability based method for discovering structure in clustered data. *Pacicific Symposium on Biocomputing* 6–17.

Gervini, D., and V. Rousson. 2004. Criteria for evaluating dimension-reducing components for multivariate data. *American Statistician* 58: 72–76.

Gorst-Rasmussen, A. 2011. *tt*. Stata add-on for performing treelet transformation. URL `http://people.math.aau.dk/ gorst/software.htm.`

Gorst-Rasmussen, A., C. C. Dahm, C. Dethlefsen, T. Scheike, and K. Overvad. 2011. Exploring dietary patterns by using the treelet transform. *American Journal of Epidemiology* 173: 1097–1104.

Jolliffe, I. T. 2002. *Principal Components Analysis*. 2nd ed. New York: Springer.

Lee, A. B., B. Nadler, and L. Wasserman. 2008. Treelets - an adaptive multi-scale basis for sparse unordered data. *Annals of Applied Statistics* 2: 435–471.

Liu, D. 2010. *treelet: Treelet*. R-package version 0.2-0, URL http://cran.r-project.org/package=treelet.

Zou, H., T. Hastie, and R. Tibshirani. 2006. Sparse principal component analysis. *Journal of Computational and Graphical Statistics* 15: 265–286.